Distributed Algorithms for Finding Central Paths in Tree Networks

ESTHER JENNINGS

Computer Science Department, California State Polytechnic University, Pomona
Email: ehjennings@csupomona.edu

Given a graph $G = (V, E)$ and a path P, let $d(v, P)$ be the distance from vertex $v \in V$ to path P. A path-center is a path which minimizes the eccentricity $e(P) = \max_{v \in V} d(v, P)$ such that for every path P' in G, $e(P) \le e(P')$, and for every subpath $P'' \subset P$, $e(P) < e(P'')$. Similarly, a core is a path which minimizes the distance $d(P) = \sum_{v \in V} d(v, P)$ such that for every path P' in G, $d(P) \le d(P')$, and for every subpath $P'' \subset P$, $d(P) < d(P'')$. We present distributed algorithms for finding path-centers and cores in asynchronous tree networks. We then extend these to compute a subset-centrum path in trees which minimizes the distance with respect to a subset of vertices, $d_S(v, P) = \sum_{v \in S} d(v, P)$, where $S \subseteq V$. The time and communication complexities of these algorithms are $O(D)$ and $O(n)$ respectively, where D is the diameter and n is the number of vertices (edges) of the network. These algorithms are asymptotically optimal.

Received 17 April, 1997; revised 30 August, 1999

1. INTRODUCTION

Given a network, the location problem concerns how to choose the 'best' sites to set up certain facilities to serve the other sites. In general, one would like to set up a facility at a 'central' location. Two extensively studied criteria for centrality are: *minimax* (centers) and *minisum* (medians) [1, 2, 3]. A *center* of a graph is a vertex c such that the maximum distance from c to any other vertex is minimized. A *median* of a graph is a vertex m such that the maximum sum of distances from m to all the other vertices is minimized. Alternatively, a median m minimizes the average distance from m to all the other vertices. So, if the goal is to have fast response from the facility, then we should locate the facility at the center of the network. However, if the main concern is the total communication cost to all the sites, then we should locate the facility at the median of the network.

The concepts of centers and medians have been extended to path-centers and cores, where facilities are located along a path. Given a graph $G = (V, E)$ and a path P, let $d(v, P)$ be the distance from vertex $v \in V$ to path P. In [4], Slater defined the *path-center* of a graph G as a path P with minimum eccentricity. Formally, a path-center is a path which minimizes the eccentricity $e(P) = \max_{v \in V} d(v, P)$ such that for every path P' in G, $e(P) \le e(P')$, and for every subpath $P'' \subset P$, $e(P) < e(P'')$. In [5], Morgan and Slater defined the *core* of a graph G as the path P which minimizes the total distance from all vertices of G to P. Formally, a core is a path which minimizes the distance $d(P) = \sum_{v \in V} d(v, P)$ such that for every path P' in G, $d(P) \le d(P')$, and for every subpath $P'' \subset P$, $d(P) < d(P'')$. Linear-time sequential algorithms for finding path-centers and cores in trees were given in [4] and [5] respectively. However, no distributed algorithms for these problems are known to date. Note that the path-center and core problems are NP-complete in general graphs because we can reduce the Hamiltonian path problem to these problems [6].

In this paper, we apply the *basic tree technique* of [7] to obtain distributed algorithms for finding path-centers and cores in trees. Considering the possibility of multiple initiators in the algorithm of [7] the basic tree technique uses at most $2n + s - 2$ messages and $\lceil 3D/2 \rceil$ time, where n is the number of vertices, D is the diameter, and s is the number of initiators. We also extend the concept of cores to the subset-centrum path, and present a distributed algorithm for solving this problem. The time and communication complexities of these algorithms are $O(D)$ and $O(n)$ respectively; the algorithms are asymptotically optimal.

We consider the standard model of an asynchronous point-to-point communication network (e.g. [8]). The network is represented by an undirected simple graph on n vertices (processors) and ℓ edges (bidirectional communication links). Each processor has a unique identity. There is no shared memory and processors communicate by message passing. Each message contains $O(\log n)$ bits. For the sake of time complexity analysis, we assume that each message incurs a delay of at most one unit of time [9]. Note that the delay assumption is only used to *estimate* the performance of our algorithms. This does not imply that our model is synchronous, neither does it affect the correctness of our algorithms. That is, our algorithms work correctly even in the absence of this delay assumption.

In Sections 2, 3 and 4, we present our algorithms for finding path-centers, cores and subset-centrum paths respectively. Section 5 concludes the paper.

2. PATH-CENTER

An important observation in [4, Theorem 4] states that the path-center of a tree must contain the center. Therefore, we propose the following two-step algorithm for finding a path-center Q:

(i) Locate the center of T using the center finding algorithm in [7].
(ii) From the center, extend a maximal path P with minimum eccentricity. Then shorten P to obtain the minimal sub-path Q of P such that $d(Q) = d(P)$.

To extend a maximal path P from the center such that $d(P)$ is minimized, each vertex must store the maximum distances it received from its neighbors during center finding. From the stored information, a center can easily determine whether it is the only center or one of the two centers. Let T be rooted at the center (break ties deterministically using processor identities). If there is only one center c, then c will extend P in two directions, to the children from which the two largest distances were received. If there are two centers, then P contains both centers and the edge connecting them; then, each of the centers will extend P in one direction, to the child (not the other center) from which the largest distance was received. From the center(s), ⟨**extend**⟩ messages are sent to extend P. Each vertex v receiving an ⟨**extend**⟩ message on edge e marks e as belonging to P. If v is not a leaf, then it propagates an ⟨**extend**⟩ message to the child from which it received the largest distance. If several choices are possible, then break ties deterministically. In order to activate all the leaves to start the computation of $Q \subset P$, as a vertex sends its ⟨**extend**⟩ message to a child, it also sends an ⟨**activate**⟩ message to each of its other children.

The computation of Q is started by the leaves when they have received either an ⟨**extend**⟩ or an ⟨**activate**⟩ message. Maximum distance information with respect to P is gathered from the leaves. While traversing toward the center, an internal vertex $v \notin P$ increments the largest of all received distances by one. A vertex on P will merely pass the maximum distance it received. When the root of T receives the maximum distance, it sends an ⟨**short**, *dist*⟩ message on P toward the leaves, where *dist* is the maximum distance received with respect to P. It is obvious that we can shorten P from both ends by *dist* edges without increasing $e(P)$. The shortened path $Q \subseteq P$ is the path-center. The propagation of ⟨**short**, *dist*⟩ stops when a vertex of P at *dist* from a leaf is reached. A vertex knows its distance from the leaves because this information is stored previously. Every vertex which received a ⟨**short**, *dist*⟩ message on edge e will mark e as belonging to the path-center Q.

THEOREM 2.1. *A path-center of a tree $T = (V, E)$ can be found in $\lceil 7D/2 \rceil$ time using $4n + 3D/2 + s$ messages, where $|V| = n$, D is the diameter of T, and s is the number of initiators.*

Proof. From [7], the center(s) of a tree T can be found by using at most $2n + D/2 + s - 2$ messages in $\lceil 3D/2 \rceil$ time, where n is the number of vertices, D is the diameter, and s is the number of initiators.

A maximal path P with minimum eccentricity can be found using $D/2$ time and $n - 1$ messages. Suppose $e(P)$ is not minimum, then there is a path R such that $e(R) < e(P)$. Let c be the center(s) of T. From [10, Lemma 3.1], $c \in P$ and $c \in R$. The above imply that there exists a vertex v on the path between c and one of the leaves of R such that $d(v, p) < d(v, r)$, where p and r are end-points of P and R respectively, and both p and r are on the same side of v with respect to c. Then, in our algorithm, v would choose to extend the path toward r instead of toward p. Since the algorithm yields a path P, this implies that such a vertex as v does not exist. Therefore, a path R with $e(R) < e(P)$ does not exist, contradicting our assumption. The time required is $D/2$ because messages are sent from the center(s) of T in two directions simultaneously.

Knowing P, the path-center Q can be found using $n-1+D-2e(P)$ messages and $D+D/2-e(P)$ time. Since $Q \subseteq P$ and $e(P)$ is minimum, $e(Q)$ is also minimum. Furthermore, Q is the minimal sub-path where $e(Q) = e(P)$ because the distance from the end-points of Q to the leaves is equal to $e(P)$. The gathering of maximum distances with respect to P requires $n - 1$ messages and D time. For the marking of Q, $D - 2e(P)$ messages are sent. Since this is done on Q in both directions from the root (a center) of T, the time required is $D/2 - e(P)$. □

3. CORE

In contrast to a path-center, a core is not necessarily unique and does not necessarily pass through a median of T [5]. Therefore, knowing the median does not help. In [5], Morgan and Slater presented a linear-time sequential algorithm which computes a core of tree networks. Optimal parallel algorithms for the computation of a core of tree networks using tree contraction techniques have been presented in [11, 12]; these algorithms use $O(n/\log n)$ processors and $O(\log n)$ time on the EREW PRAM. The main observation used in these algorithms is that an end-point v of a core can be identified by finding a *rooted core*.

We combine this observation with the basic tree technique to obtain an asynchronous distributed algorithm for finding cores of tree networks in $O(D)$ time using $O(n)$ messages where D is the diameter and n is the number of vertices of the network. We assume that all network edges have unit weight. Our algorithm may have multiple initiators. When the algorithm terminates, every vertex knows whether it is a vertex of the computed core P and which of its incident edges are in P. The algorithm can be easily modified to find a core in a tree network where edges have arbitrary positive weights. Lemma 3.1 states the effect on the sum of distances as a path P is being extended to include more vertices.

LEMMA 3.1. [5, Section 2] *Let v be a vertex on a core P where $|P| > 1$, and let the vertices adjacent to v be $u_1, u_2, \ldots, u_{\delta(v)}$. Let T_{u_i} denote the subtree rooted at u_i not containing v. The following conditions hold for a vertex*

$w \in T_{u_i}$, $1 \le i \le \delta(v)$: (1) *If* $u_i \notin P$, *then* $d(w, P) = d(w, v)$. (2) *If* $u_i \in P$, *then* $d(w, P) < d(w, v)$.

That is, if P is extended from v to u_i, $u_i \notin P$, then the distances from vertices $w \in T_{u_i}$ to P will decrease. However, an arbitrary choice of u_i does not guarantee that the extended path is a core. Let $\varrho_{v\to u}$ denote the path from v to u in T. In [11, Lemma 1], the total decrease in sum of distances from $w \in T_{u_i}$ to P (extended from v) is called the *reduction*, designated as *reduc*($\varrho_{v\to u_i}$).

DEFINITION 3.1. [11, Lemma 1] (reduction) *Let Q be a path ending at s. The total reduction in $d(Q)$ by extending Q to include the vertices of $\varrho_{s\to v}$ is reduc($\varrho_{s\to v}$) $= \sum |T_w|$, where w is a vertex on $\varrho_{s\to v}$ excluding s, and T_w consists of w and those subtrees rooted at the neighbors of w which do not contain s.*

LEMMA 3.2. (1) *A core connects two leaves of T* [12, Lemma 1]. (2) *Let P_r be a rooted core, i.e. a path containing r which minimizes the sum of distances from other vertices to P_r. Then a core and P_r must share at least one common vertex* [12, Lemma 2]. (3) *Let tree T be rooted at an arbitrary vertex r. A vertex v is in a core if reduc($\varrho_{r\to v}$) the maximum possible value* [11, Lemma 3].

Note that there may be several cores of T, so that the choice of v in Lemma 3.2(3) may not be unique. To find a core of a tree network, we first root the network T at an arbitrary vertex r. Then we find an end-point v of a core P of T by using Lemma 3.2(3). Once v is found, we extend a core P from v so that maximum reduction is achieved. Since more than one core is possible, we break ties arbitrarily. The algorithms described in Lemmas 3.3 and 3.4 follow the algorithm of [5], except we combine their method with the basic tree technique of [7] to obtain a distributed algorithm. The non-trivial part is that the complexity of our distributed core algorithm is less than the complexities of simply running a distributed rooted core algorithm twice.

LEMMA 3.3. *Given a tree network $T = (V, E)$, an end-point v of a core is known at a saturated vertex within $\lceil 3D/2\rceil$ time using at most $2n + s - 2$ messages, where $n = |V|$, D is the diameter of T, and s is the number of initiators.*

Proof. We apply the basic tree technique to define a root r which is a saturated vertex. During the backward phase, the size of the traversed subtree and the largest reduction obtainable by extending a path into that subtree are reported. When a leaf ℓ is activated, it sends a $\langle$**parent**, $|T_\ell|$, *reduc* $= 1\rangle$ message to its only neighbor. When an internal vertex u has received a $\langle$**parent**, $|T_i|$, *reduc*$_i\rangle$ message from all but one of its edges, it marks the edge on which the maximum *reduc*$_i$ is received. If the maximum reduction is reported on several edges, then mark one of these edges arbitrarily. Vertex u also computes the size of its subtree $|T_u| = 1 + \sum |T_i|$, and its reduction *reduc*($\varrho_{u\to\ell}$) $= |T_u| + \max($*reduc*$_i)$, where i are the children of u. Then u sends a $\langle$**parent**, $|T_u|$, *reduc*($\varrho_{u\to\ell}$)$\rangle$ on the edge from which no $\langle$**parent**, *size*, *reduc*$\rangle$ message was received, if such an edge exists. From [10, Lemma 3.1], either one or two vertices become saturated. If there is only one saturated vertex, then it is chosen as the root r. Otherwise, there are exactly two adjacent saturated vertices contending to be the root. Ties are then broken by unique processor identities; for example, the processor with the smaller identity wins.

At vertex r, r knows which of its incident edges leads to a vertex v such that *reduc*($\varrho_{r\to v}$) is maximum. This marked path with the maximal reduction satisfies Lemma 3.2(3), so an end-point v of a core of T is computed correctly at r. Since we are merely using the basic tree technique to gather information concerning the sizes of subtrees and reductions, the time and messages used are $\lceil 3D/2\rceil$ and $2n + s - 2$ respectively. A message size of $O(\log n)$ bits suffices because the size of a subtree is bounded by n, and reduction is bounded by $n^2/2$. □

Let T_r denote the tree network rooted at r. Note that any vertex u of T_r knows the maximum reduction obtainable from each of its children after saturation at r. The only piece of information not available at u is how much reduction it can obtain from the direction of its parent. From the root r of T_r, we have a marked path leading to v where *reduc*($\varrho_{r\to v}$) is maximum. The root r will then send a message along the marked path to saturate every vertex between r and v. The size of subtree and reduction (from the direction of r) are updated at each vertex along the path and are passed with the message as it propagates toward v. The saturation of v can be regarded as re-rooting the tree network at v because by the time v becomes saturated, every vertex in T_v knows the reduction obtainable from each of its children in T_v.

LEMMA 3.4. *Given an end-point v of a core P of a tree T, the other end-point v' of P can be found by finding a vertex v' such that reduc($\varrho_{v\to v'}$) is maximized.*

Proof. Given an end-point v of a core P of T, and by the definition of a core, the other end-point v' of P (which by Lemma 3.2(1) must be a leaf) can be found by extending a path from v in the direction which maximizes *reduc*($\varrho_{v\to v'}$). Then, the unique path connecting v to v' is optimal as a core. Note that the choice of v' might not be unique. □

THEOREM 3.1. *Given a tree $T = (V, E)$, a core of T can be computed by exchanging $2n + 2D + s - 2$ messages in $3D$ time, where $n = |V|$, D is the diameter of T, and s is the number of initiators.*

Proof. The correctness of the algorithm follows from Lemma 3.2 and Lemmas 3.3, 3.4. From Lemma 3.3, the first end-point v of a core $P \in T$ is known at a saturated vertex r within $\lceil 3D/2\rceil$ time by exchanging at most $2n + s - 2$ messages. Suppose r is saturated at time $k + D$ after the algorithm starts, where $k \le \lceil D/2\rceil$. Then $d(r, v)$ is bounded by $D - k$, so the backward phase of the basic tree technique and the propagating of messages to re-root T at v takes at most D time altogether. That is, the end-point v of a core is saturated within $2D$ time after the algorithm starts. The number of messages sent from r to saturate v is bounded by D. To trace a core from v to v' requires at most an additional D messages and time. Summing these, a core P of a tree network can be computed within $3D$ time using at most $2n + 2D + s - 2$ messages. □

4. GENERALIZATION TO SUBSET-CENTRUM PATH

Suppose we are not interested in minimizing the total distance from a facility to all the sites, but in minimizing the total distance from the facility to certain important sites. We define the S-distance of a path P as $d_S(P) = \sum_{v \in S} d(v, P)$. Given a graph $G = (V, E)$ and a subset S of vertices, where $|S| = k$, the k-centrum path is defined as a path P such that $d_S(P)$ is a minimum. Intuitively, each vertex v of S contributes a distance of $d(v, P)$ to $d_S(P)$; vertices outside of S contribute nothing to $d_S(P)$.

Given a tree network $T = (V, E)$ and a subset $S \subseteq V$, the strategy is to first compute a minimal subtree T_S of T such that all the vertices S are contained in T_S. Then, we find a path P in T_S such that $d_S(P)$ is a minimum. Our algorithm contains two main phases. In phase one, we use the basic tree technique to define T_S. In phase two, we modify our core algorithm (Section 3) to find a k-centrum path in T_S. Obviously, the minimum subtree containing all the vertices of S must not have any leaves which are outside of S.

From a simple analysis, we obtain that phase one requires at most $2n + s - 2$ messages and $\lceil 3D/2 \rceil$ time, where $n = |V|$, D is the diameter of T and s is the number of initiators. Phase 2 is an execution of the modified core algorithm on T_S with a single initiator. The modification is in the way subtree sizes and reductions are computed. Suppose T_S has n_S vertices and diameter D_S. Then, this phase uses $2n_S + 2D_S - 1$ messages and $3D_S$ time, where $n_S \leq n$ and $D_S \leq D$, because $T_S \subseteq T$ and both T_S and T are trees. Thus, the total time and messages required are at most $4n + 2D + s - 3$ and $\lceil 9D/2 \rceil$ respectively.

5. CONCLUSION

In this paper we presented communication and time optimal asynchronous algorithms for finding central paths in tree networks. These algorithms are decentralized in the sense that any subset of vertices may start the algorithms. We first studied the path-center where the maximum distance from any vertex to the path is minimized, the *minimax* criterion. We then studied the core which minimizes the average distance from all the vertices to the core, the *minisum* criterion. We extended these to find a subset-centrum path which minimizes the total distance from the path to a subset of vertices. For future research, we intend to unify these results. Instead of a pre-specified subset of k vertices, we would like to find a k-centrum path for any subset of k vertices. More specifically, we would like to find a shortest path which minimizes the total distance from the path to any subset of k vertices in the tree. For $k = 1$, this problem is equivalent to finding a path-center. For $k = n$ (n being the number of vertices in the tree), this becomes the same as finding the core. A naive approach to solve this problem for general k is to compute all the possible subsets each containing k vertices. Then compute the subtree T_S corresponding to each subset of k vertices and apply our subset-centrum algorithm on T_S. Finally, choose the path with the minimum total distance. This approach can be costly depending on k. We seek an efficient algorithm for this problem.

ACKNOWLEDGEMENTS

This work was supported by STINT, the Swedish Foundation for International Cooperation in Research and Higher Education, under grant Dnr 97/397. Part of this work was conducted while visiting the Technion as a postdoctoral fellow.

REFERENCES

[1] Slater, P. J. (1978) Centers to centroids in graphs. *J. Graph Theory*, **2**, 209–222.

[2] Gerstel, O. and Zaks, S. (1994) A new characterization of tree medians with applications to distributed sorting. *Networks*, **24**, 23–29.

[3] Zelinka, B. (1968) Medians and peripherians of trees. *Arch. Math.*, **4**, 87–95.

[4] Slater, P. J. (1982) Locating central paths in a graph. *Transportation Sci.*, **16**, 1–18.

[5] Morgan, C. and Slater, P. (1980) A linear algorithm for a core of a tree. *J. Algorithms*, **1**, 247–258.

[6] Hakimi, S. L., Schmeichel, E. F. and Labbé, M. (1993) On locating path- or tree-shaped facilities on networks. *Networks*, **23**, 543–555.

[7] Korach E., Rotem, D. and Santoro, N. (1984) Distributed algorithms for finding centers and medians in networks. *ACM Trans. Program. Lang. Syst.*, **6**, 380–401.

[8] Gallager, R. G., Humblet, P. A. and Spira, P. M. (1983) A distributed algorithm for minimum weight spanning trees. *ACM Trans. Program. Lang. Syst.*, **5**, 66–77.

[9] Awerbuch, B. and Peleg, D. (1990) Network synchronization with polylogarithmic overhead. *31st Symp. on Foundations of Computer Science*, vol. 31, pp. 514–522. IEEE Computer Society Press, Los Alamitos, CA.

[10] Santoro, N. (1995) Distributed algorithms. *Course notes for Advanced Summer School on Distributed Algorithms and Applications*, July, Siena, Italy. University of Siena Press, Siena, Italy

[11] Albacea, E. A. (1994) Parallel algorithm for finding a core of a tree network. *Inf. Process. Lett.*, **51**, 223–226.

[12] Peng, S. and Lo, W. (1994) A simple optimal parallel algorithm for a core of a tree. *J. Parallel Distrib. Comput.*, **20**, 388–392.